


Unidade VII: Árvore Binária - Inserção



PUC Minas


Adaptação dos slides elaborados pelo Instituto de Ciências Exatas e
Informática - Departamento de Ciência da Computação

- Funcionamento básico
- Exemplo
- Inserção com retorno de referência
- Análise de complexidade

- **Funcionamento básico** 
- Exemplo
- Inserção com retorno de referência
- Análise de complexidade

Funcionamento Básico da Inserção

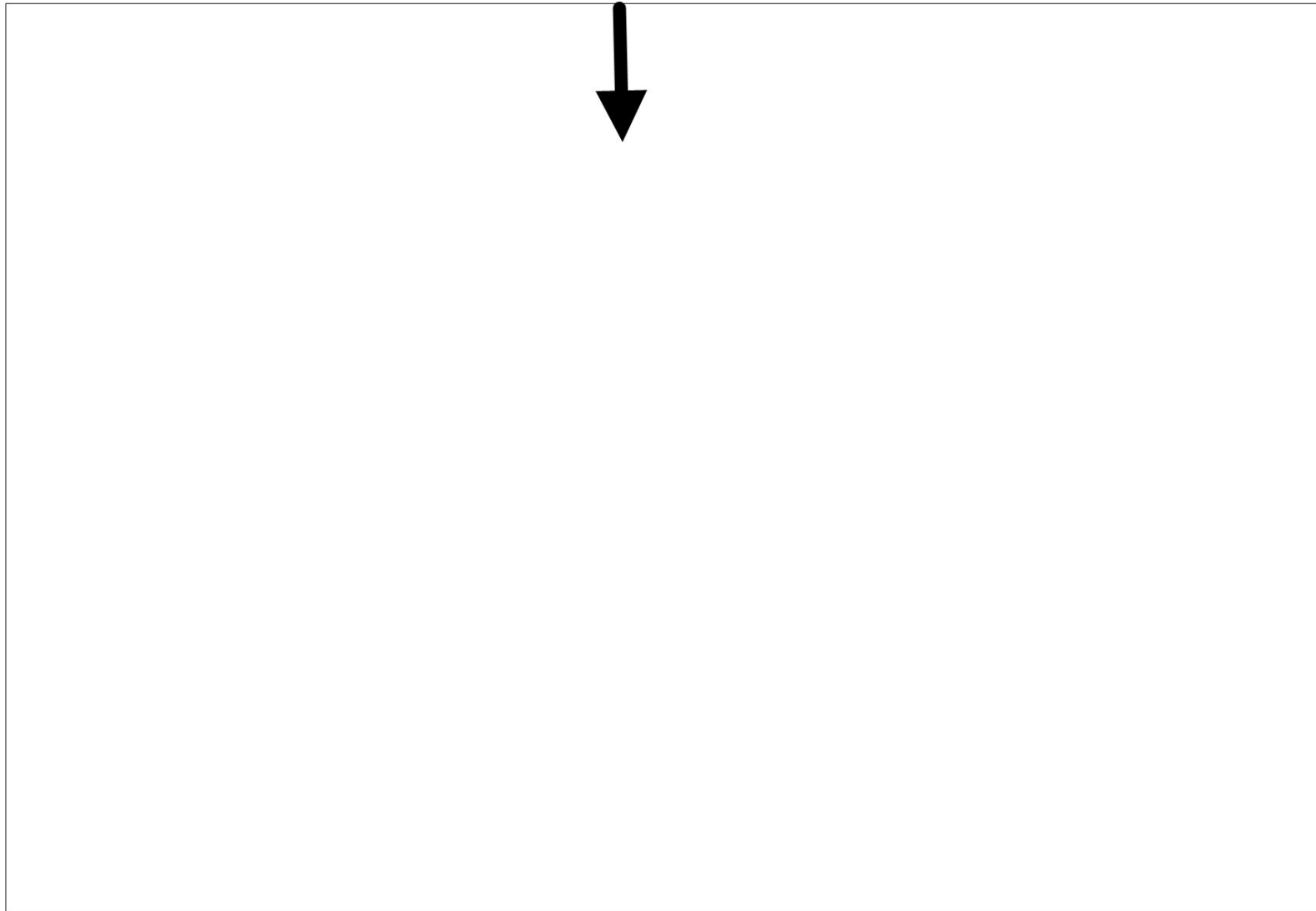
- (1) Se a raiz estiver vazia, insere-se o elemento nela
- (2) Senão, se o novo elemento for **menor** que o da raiz, chama-se recursivamente a inserção para a subárvore da **esquerda**
- (3) Senão, se o novo elemento for **maior** que o da raiz, chama-se recursivamente a inserção para a subárvore da **direita**
- (4) Senão, se o novo elemento for **igual** ao da raiz, não inserir um elemento repetido

- Funcionamento básico
- **Exemplo** 
- Inserção com retorno de referência
- Análise de complexidade

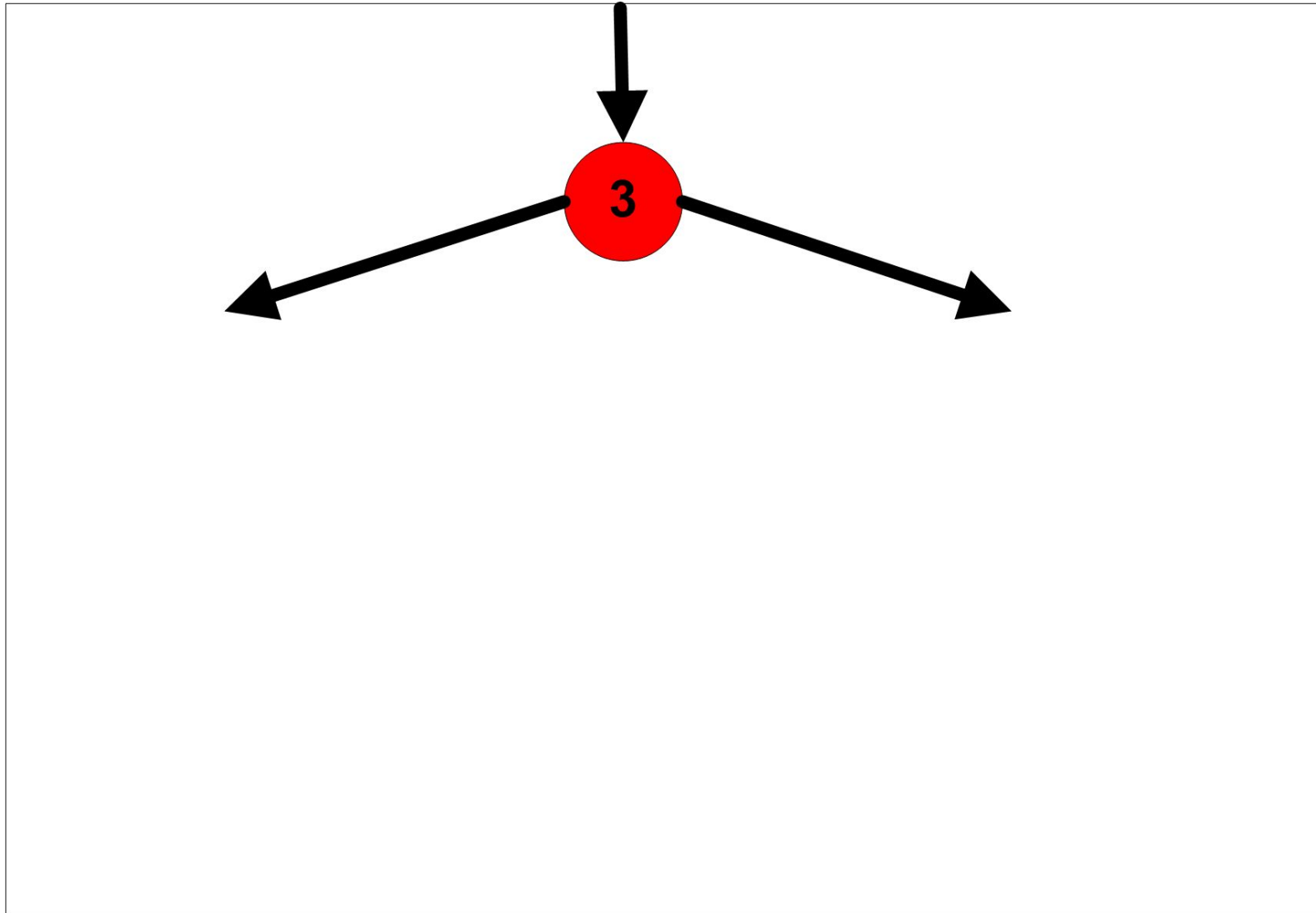
Exemplo

- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6

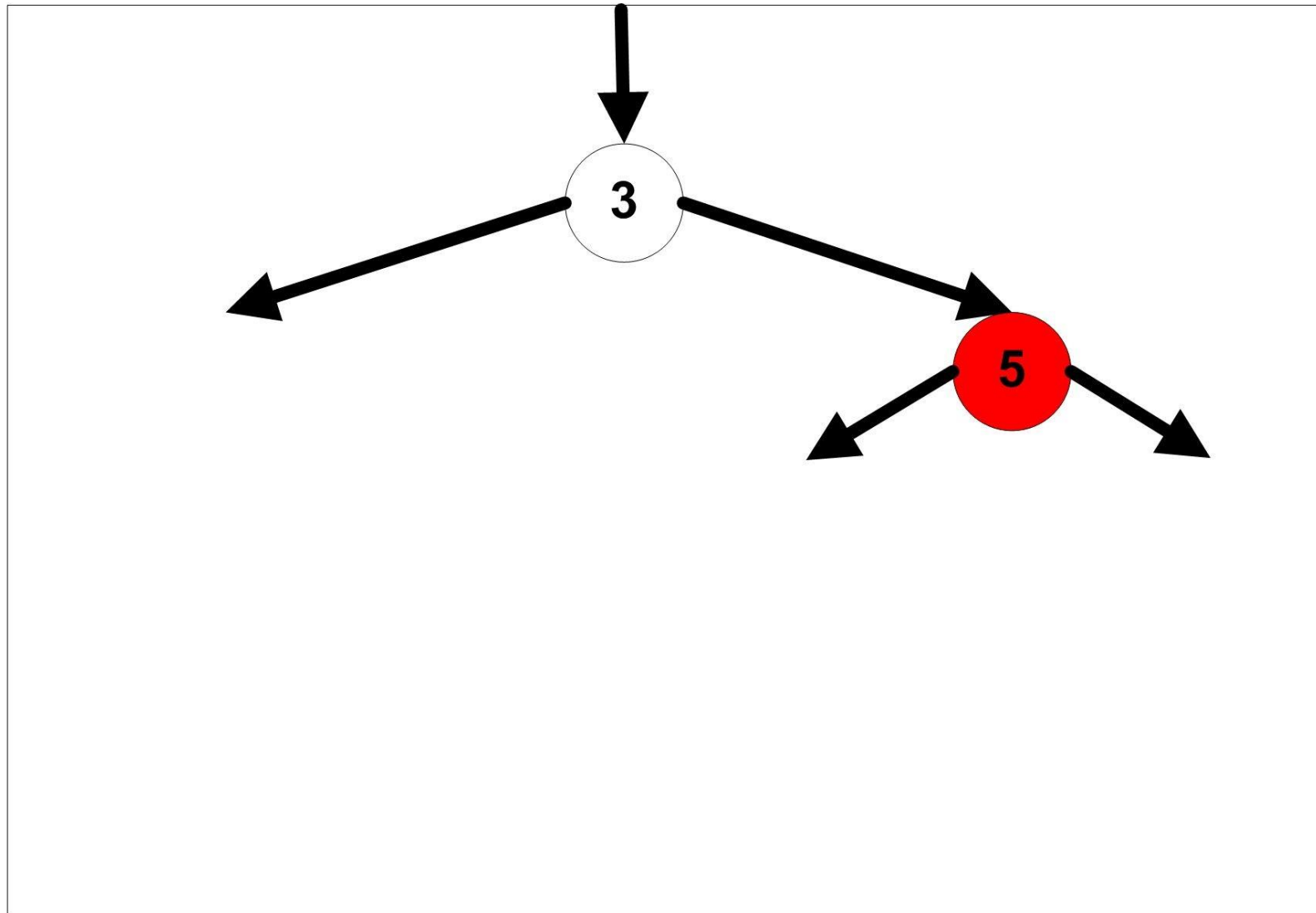
- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6



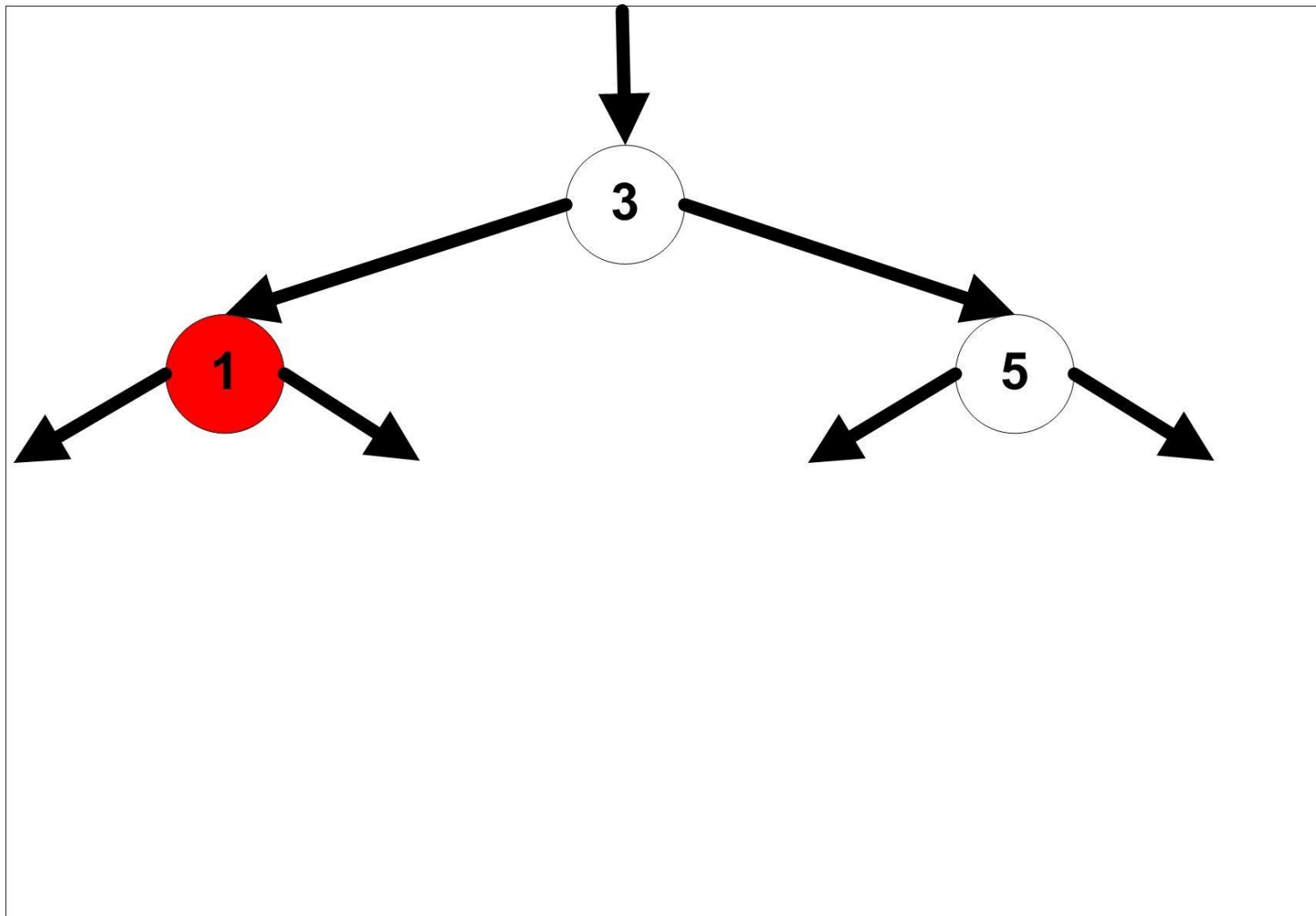
- Inserir, na ordem, os elementos **3**, 5, 1, 8, 2, 4, 7 e 6



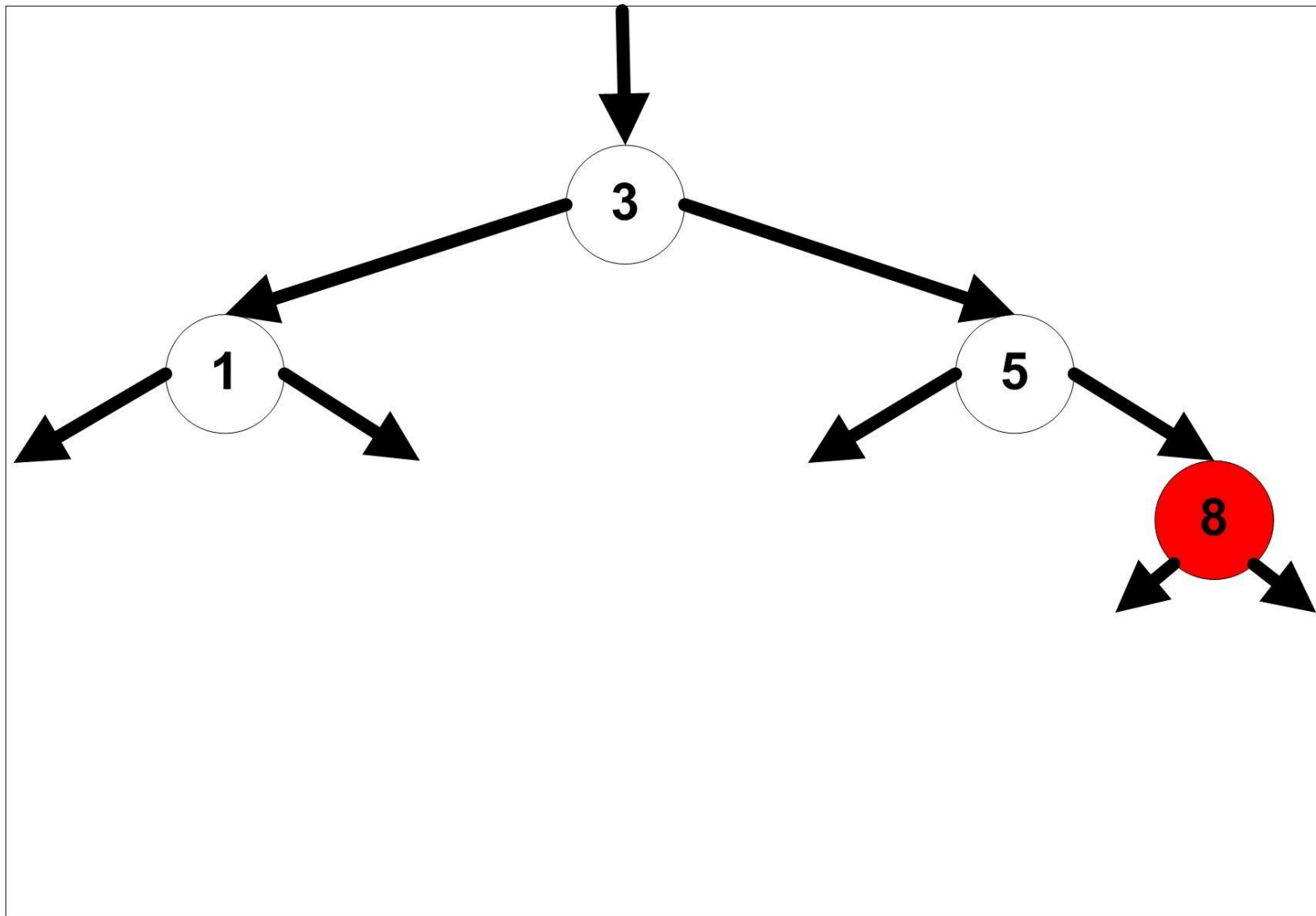
- Inserir, na ordem, os elementos 3, **5**, 1, 8, 2, 4, 7 e 6



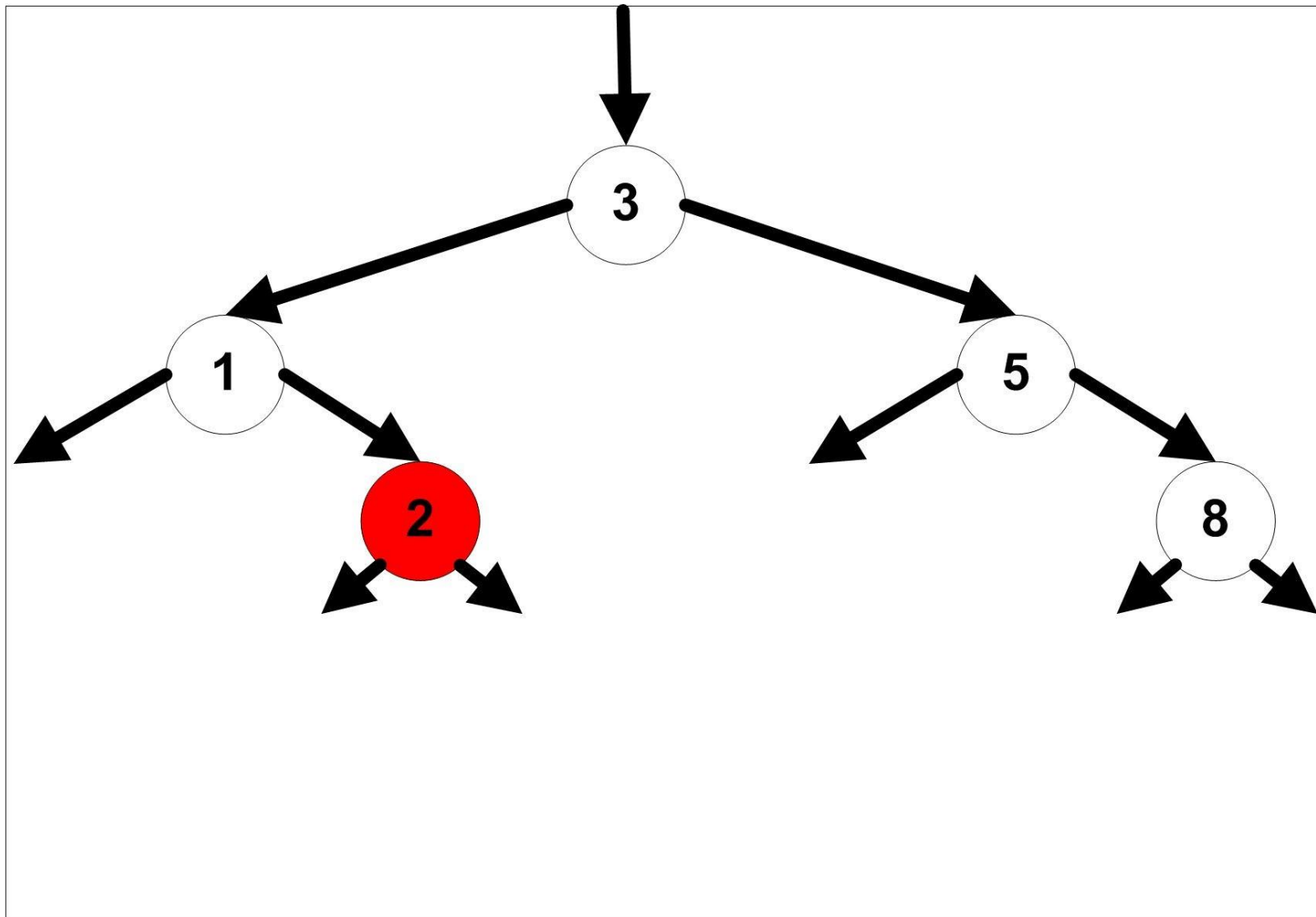
- Inserir, na ordem, os elementos 3, 5, **1**, 8, 2, 4, 7 e 6



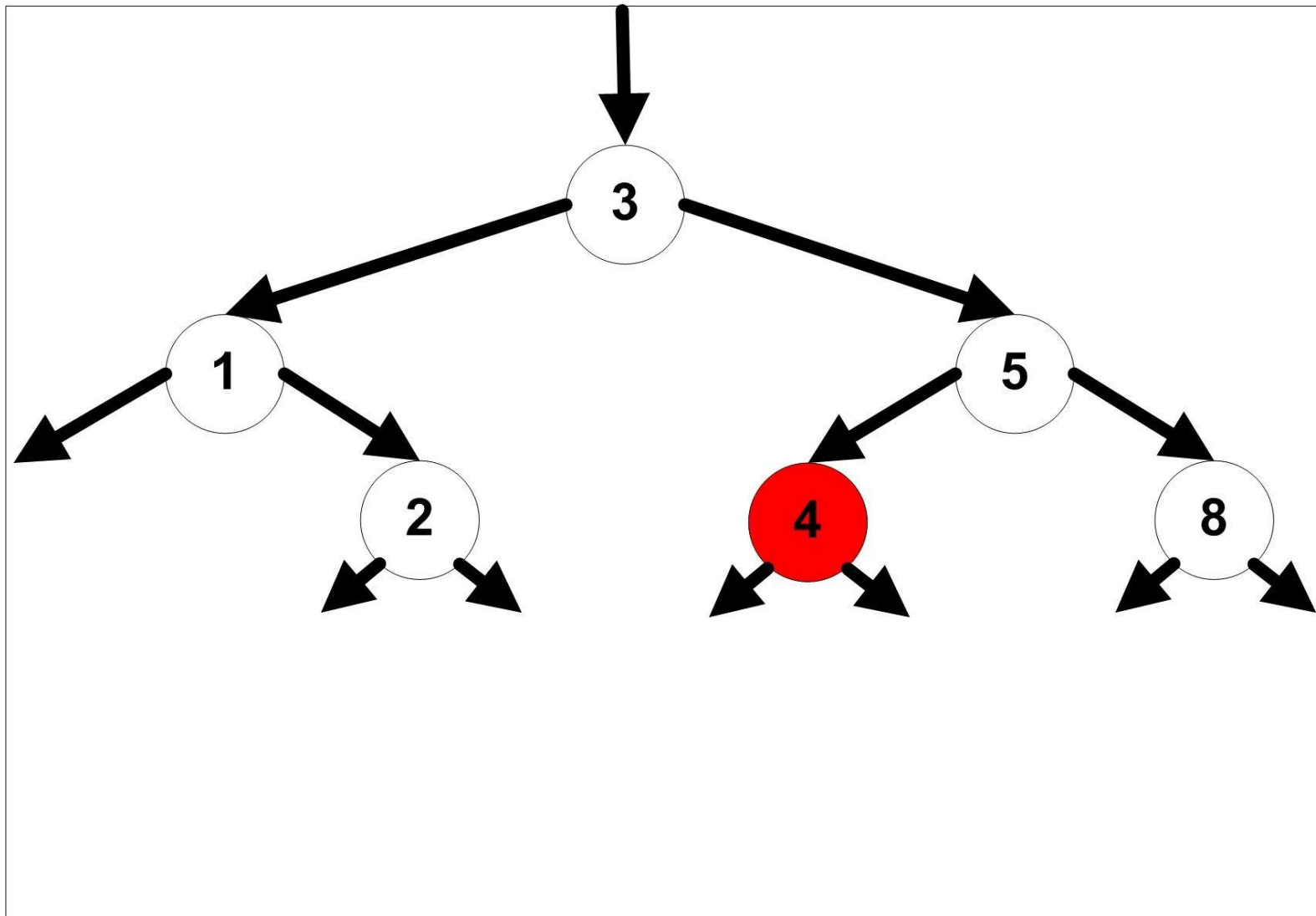
- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e 6



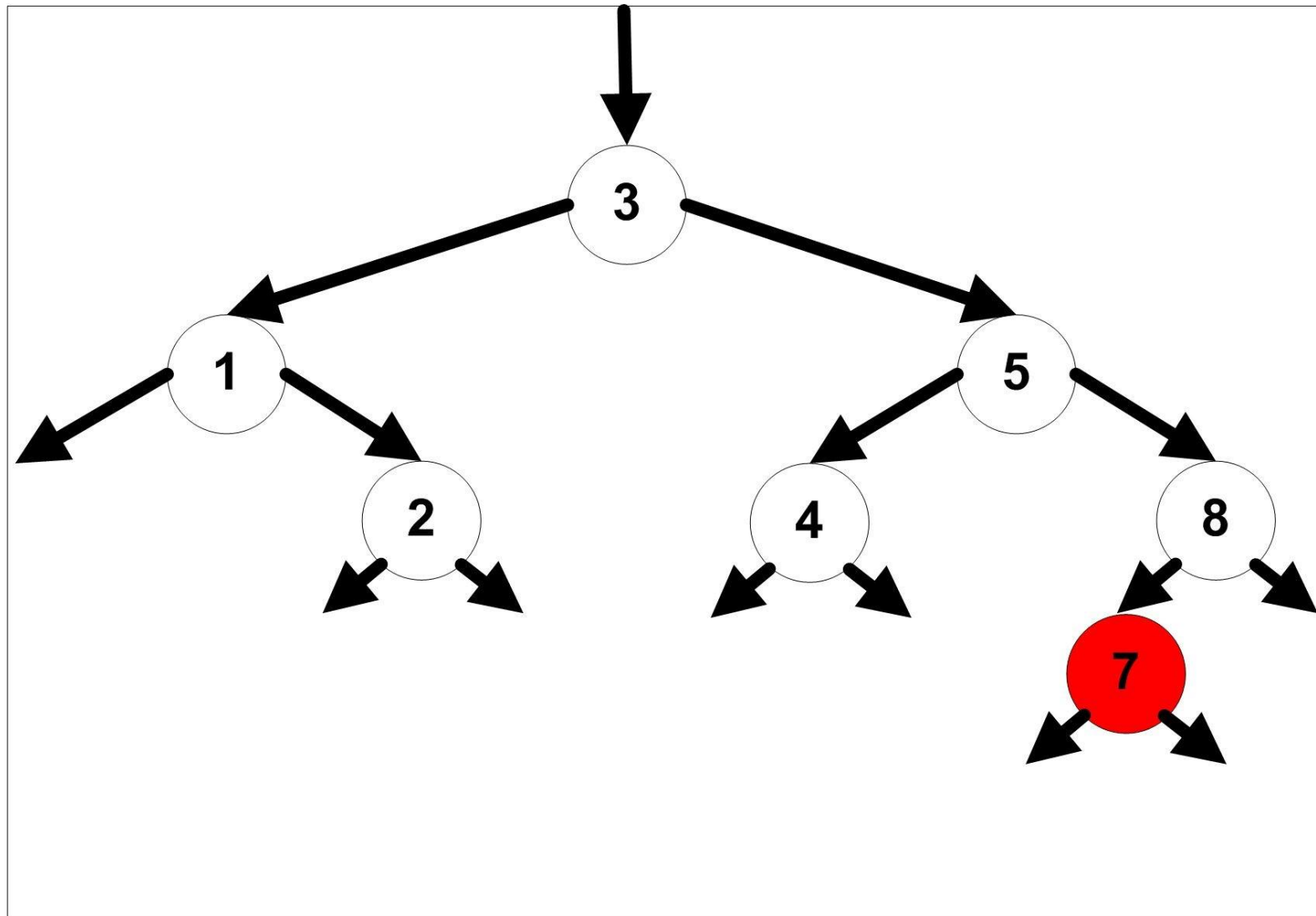
- Inserir, na ordem, os elementos 3, 5, 1, 8, **2**, 4, 7 e 6



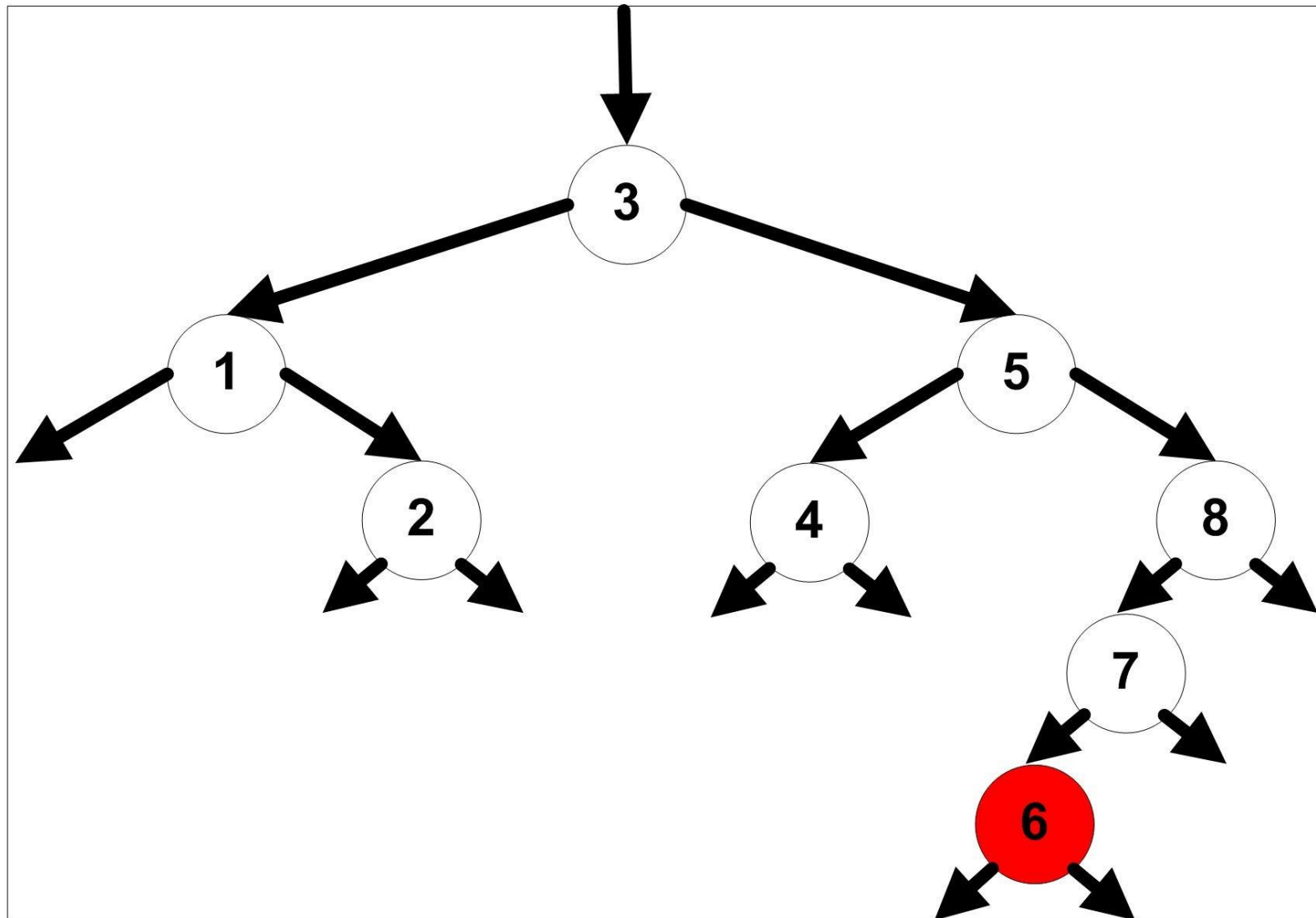
- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, **4**, 7 e 6



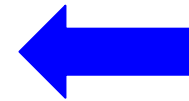
- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, **7** e 6



- Inserir, na ordem, os elementos 3, 5, 1, 8, 2, 4, 7 e **6**



- Funcionamento básico
- Exemplo
- **Inserção com retorno de referência**
- Análise de complexidade



Inserção com retorno de referência

```
class ArvoreBinaria
{
    private No raiz;
    public ArvoreBinaria(){raiz = null;}
    public void Inserir(int x) { }
    public bool Pesquisar(int x) { }
    public void CaminharCentral() { }
    public void CaminharPre() { }
    public void CaminharPos() { }
    public void Remover(int x) { }
}
```

raiz

null

Raiz



Inserção com retorno de referência

```
class ArvoreBinaria
{
    private No raiz;
    public ArvoreBinaria(){raiz = null;}
    public void Inserir(int x) { }
    public bool Pesquisar(int x) { }
    public void CaminharCentral() { }
    public void CaminharPre() { }
    public void CaminharPos() { }
    public void Remover(int x) { }
}
```

raiz null

↓ Raiz
⋮

Vamos inserir os elementos
3, 5, 1, 8, 2, 4, 7 e 6
(várias chamadas do inserir)

Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz null x 3

↓ Raiz
⋮

Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
```

```
    raiz = Inserir(x, raiz);
```

```
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```

raiz null x 3

↓ Raiz
⋮

Inserção com retorno de referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
```

raiz

null

x

3

x

3

Raiz

Cada chamada do inserir cria novas variáveis e, por isso, temos duas variáveis com nome x

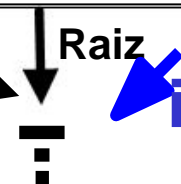
Inserção com retorno de referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new IllegalArgumentException("Elemento duplicado");
    }
    return i;
}
```



O valor inicial do ponteiro i é o mesmo do ponteiro raiz, ou seja, null

Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

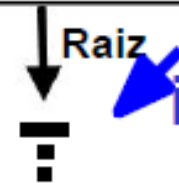
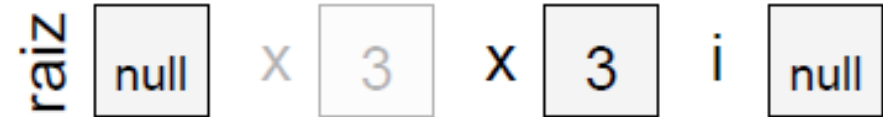
```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```

true: null == null



Inserção com retorno de referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
```

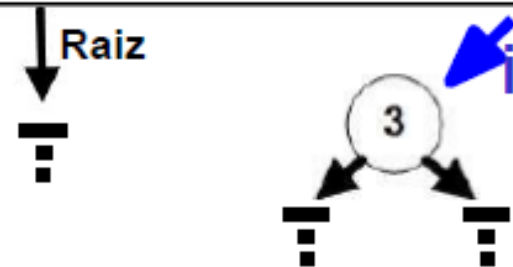
```
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
        throw new Exception("Erro!");
```

```
    }
    return i;
```

```
}
```

raiz [null] x [3] x [3] i [n(3)]



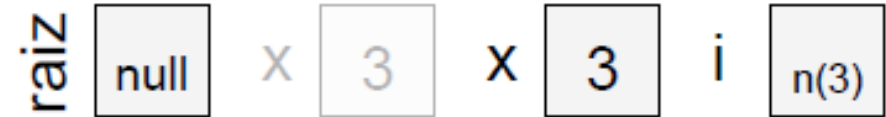
Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

retorna o endereço de n(3)

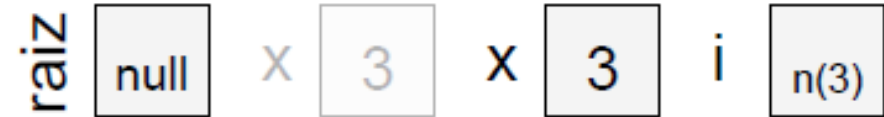


Inserção com retorno de referência

//Inserir 3, 5, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```



Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
```

```
    raiz = Inserir(x, raiz);
```

```
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

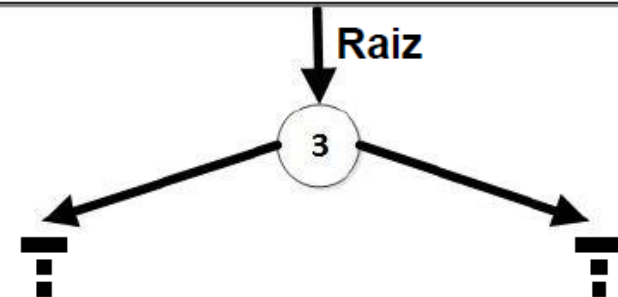
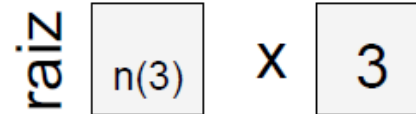
```
    } else {
```

```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```



Inserção com retorno de referência

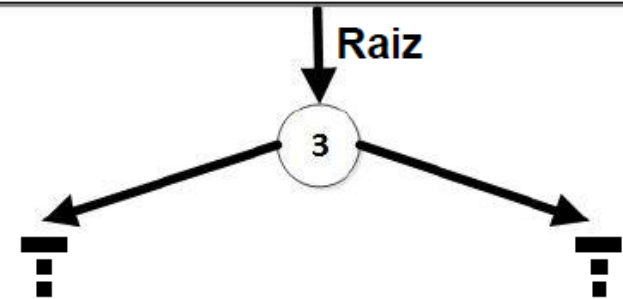
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);  
}
```

```
private No Inserir(int x, No i){  
    if (i == null){  
        i = new No(x);  
    } else if (x < i.Elemento){  
        i.Esq = Inserir(x, i.Esq);  
    } else if (x > i.Elemento){  
        i.Dir = Inserir(x, i.Dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz

n(3)

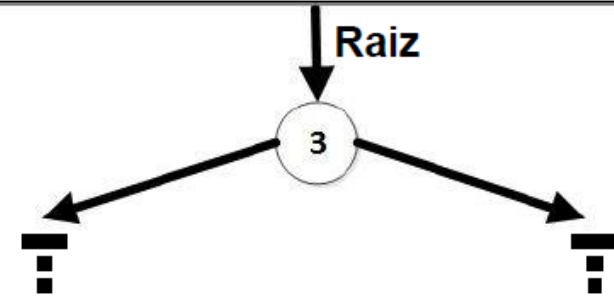


Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);  
}  
  
private No Inserir(int x, No i){  
    if (i == null){  
        i = new No(x);  
    } else if (x < i.Elemento){  
        i.Esq = Inserir(x, i.Esq);  
    } else if (x > i.Elemento){  
        i.Dir = Inserir(x, i.Dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x 5



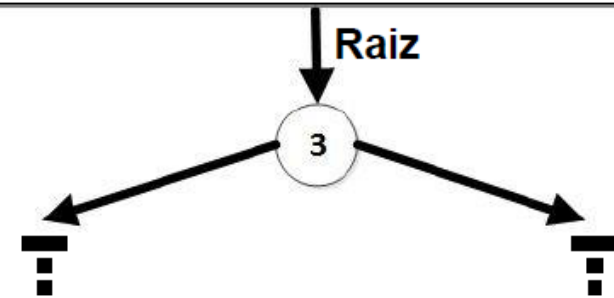
Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);  
}
```

```
private No Inserir(int x, No i){  
    if (i == null){  
        i = new No(x);  
    } else if (x < i.Elemento){  
        i.Esq = Inserir(x, i.Esq);  
    } else if (x > i.Elemento){  
        i.Dir = Inserir(x, i.Dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz n(3) x 5



Inserção com retorno de referência

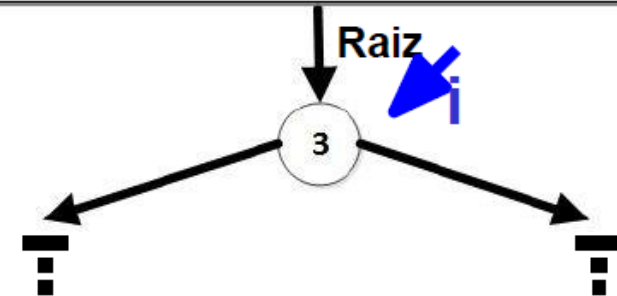
//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

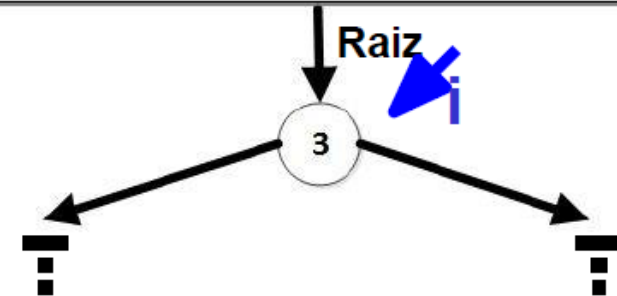
```
    }
```

```
    return i;
```

```
}
```

false: n(3) == null

raiz n(3) x 5 x 5 i n(3)



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

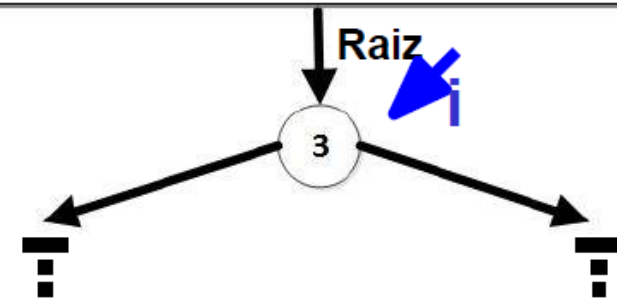
```
    }
```

```
    return i;
```

```
}
```

false: $5 < 3$

raiz n(3) x 5 x 5 i n(3)



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

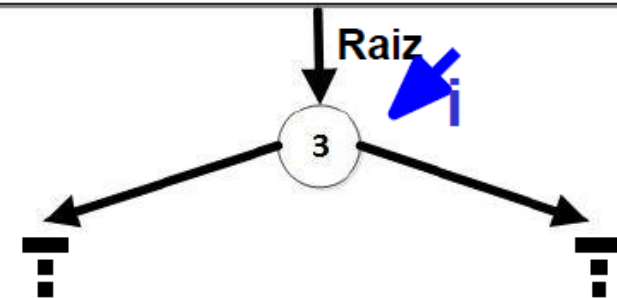
```
    }
```

```
    return i;
```

```
}
```

true: 5 > 3

raiz n(3) x 5 x 5 i n(3)



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

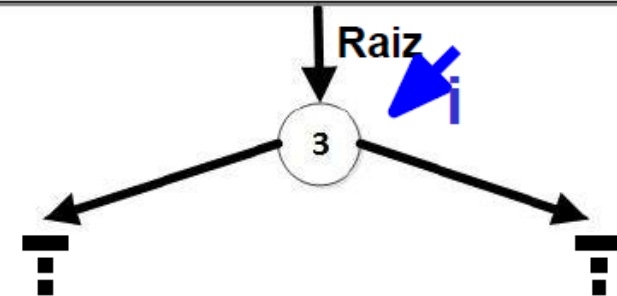
```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```

raiz n(3) x 5 x 5 i n(3)



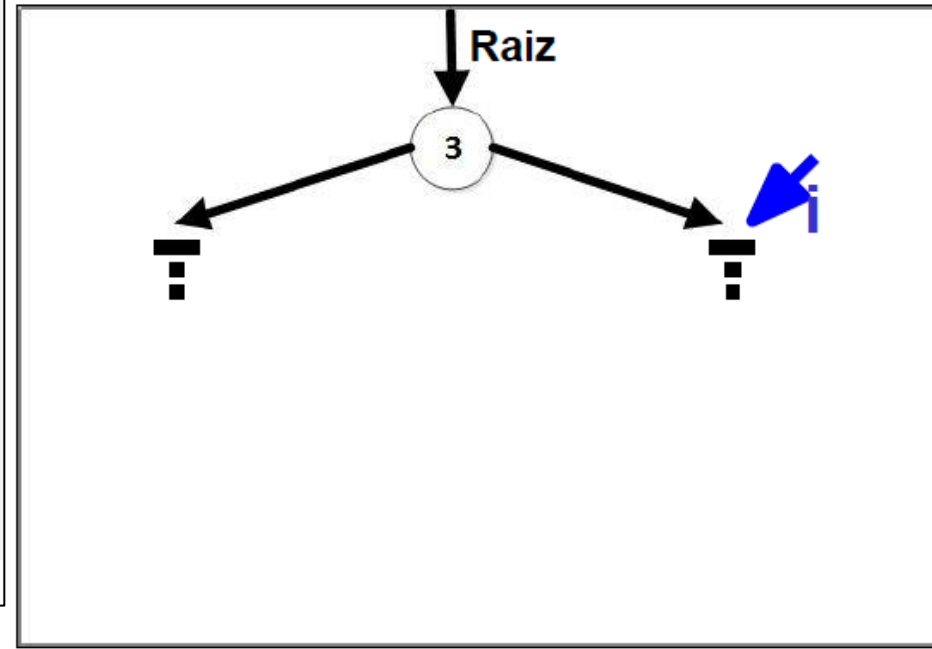
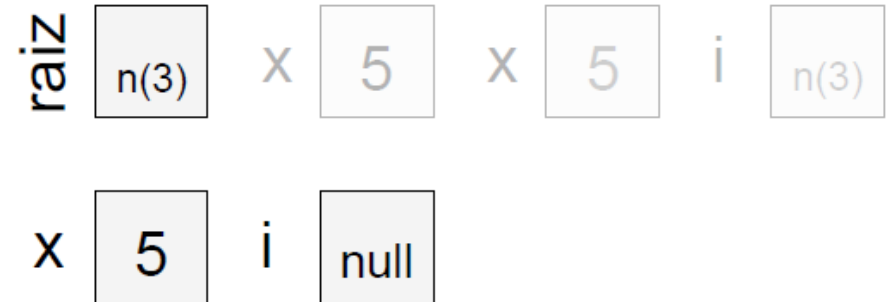
Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

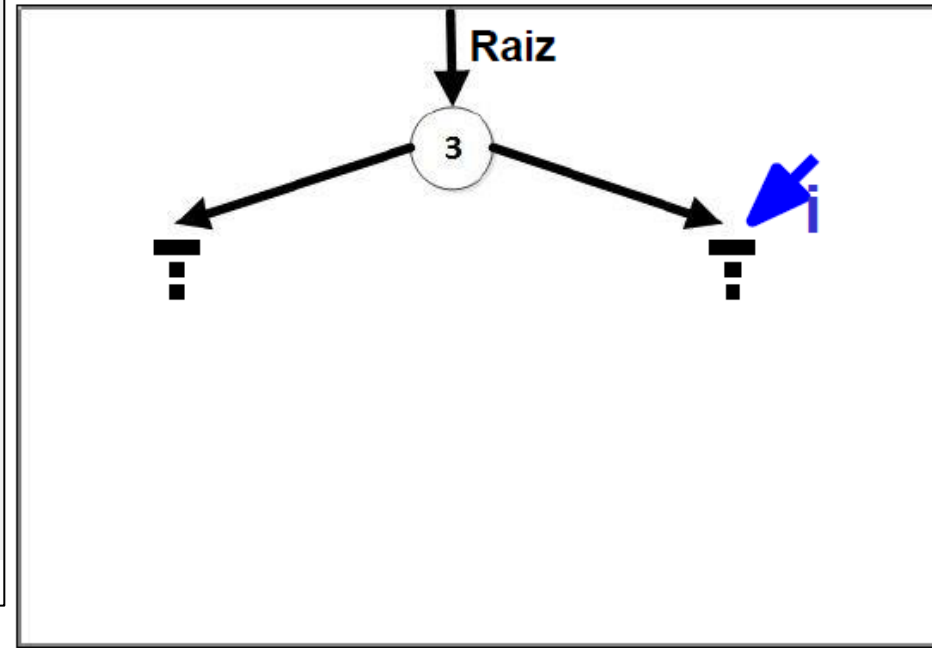
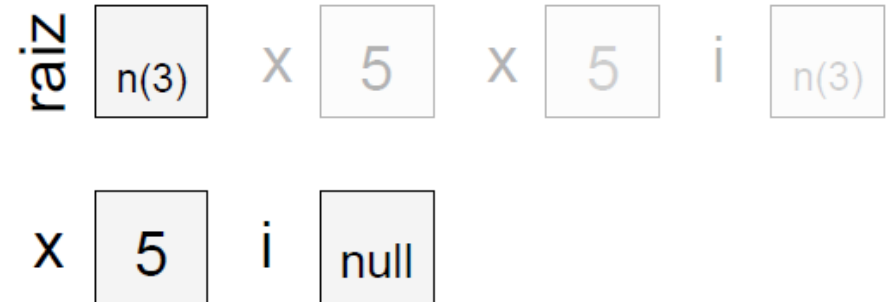
```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```

true: null == null



Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
```

```
        i = new No(x);
```

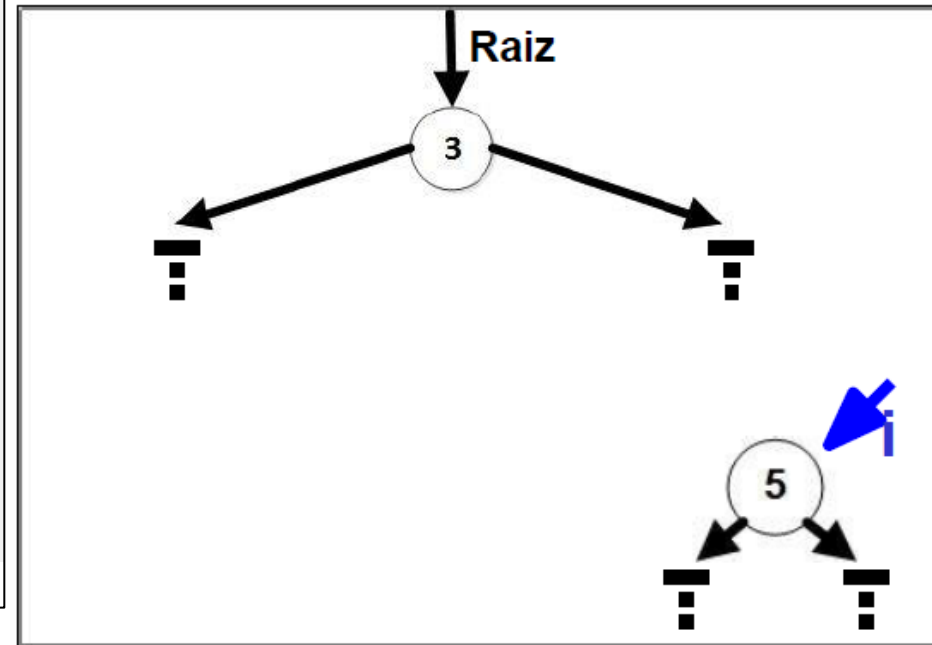
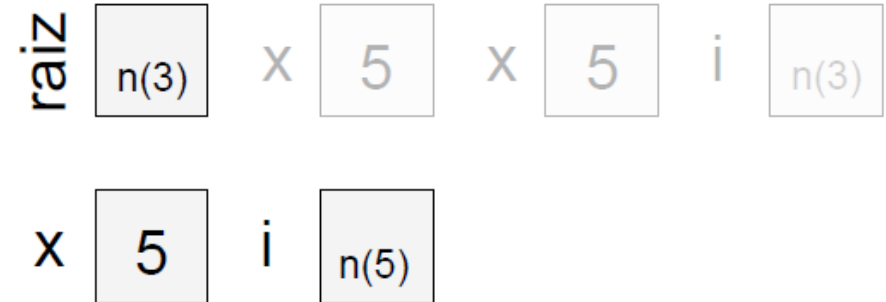
```
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
        throw new Exception("Erro!");
```

```
    }
    return i;
```

```
}
```



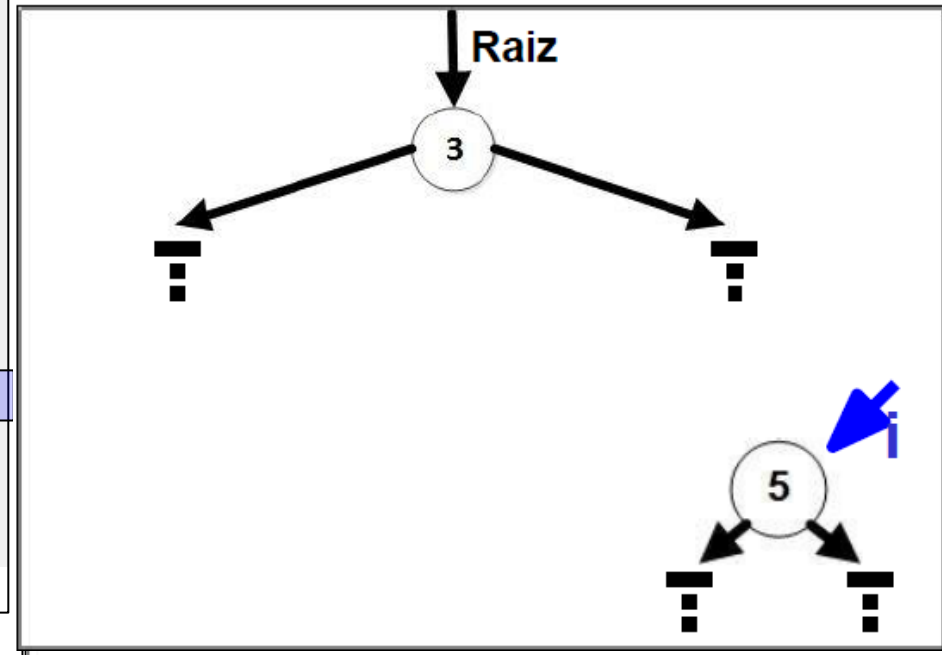
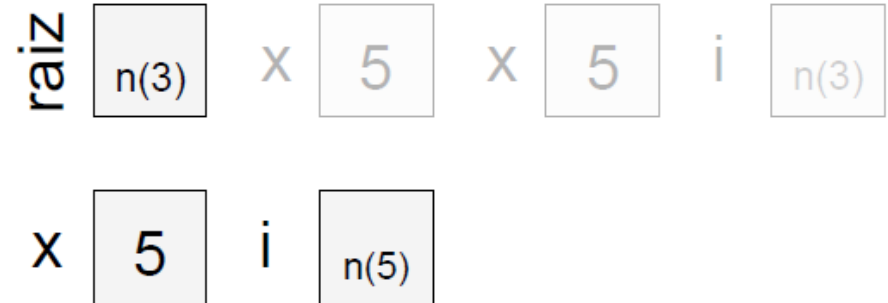
Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

retorna o endereço de n(5)

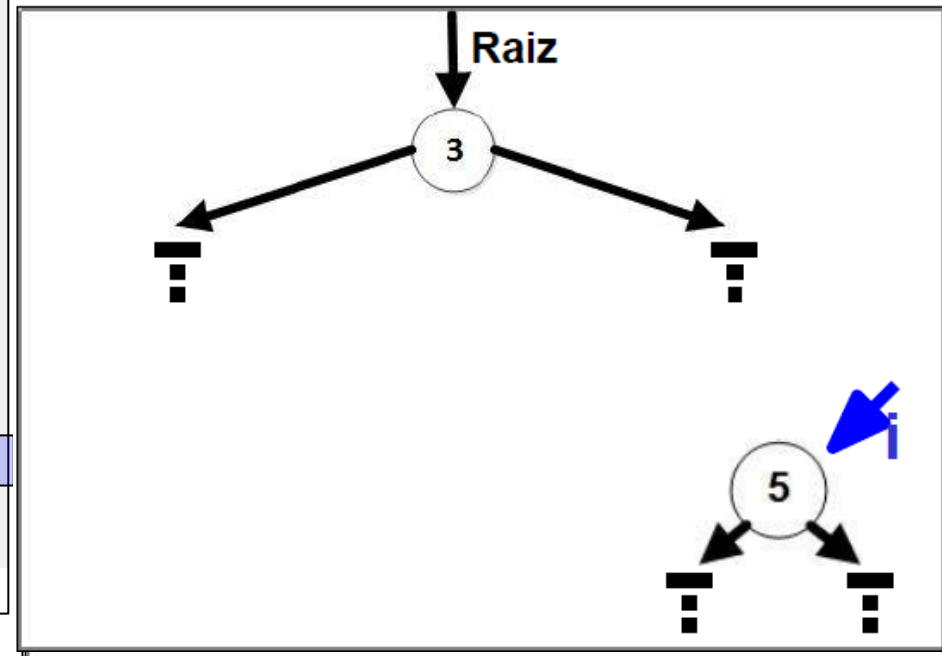
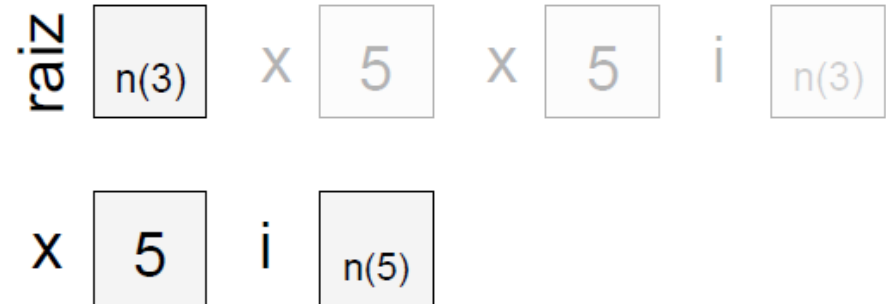


Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```



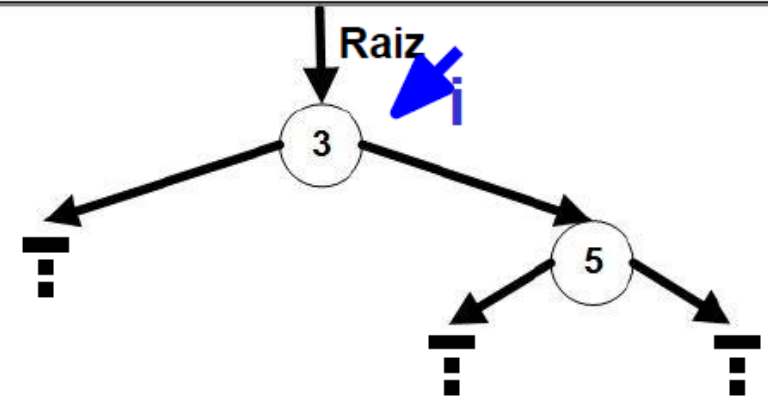
Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



Inserção com retorno de referência

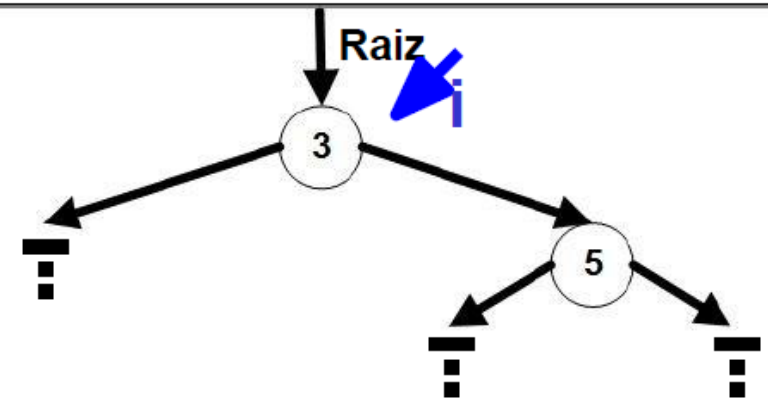
//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

retorna o endereço de n(3)

raiz n(3) x 5 x 5 i n(3)



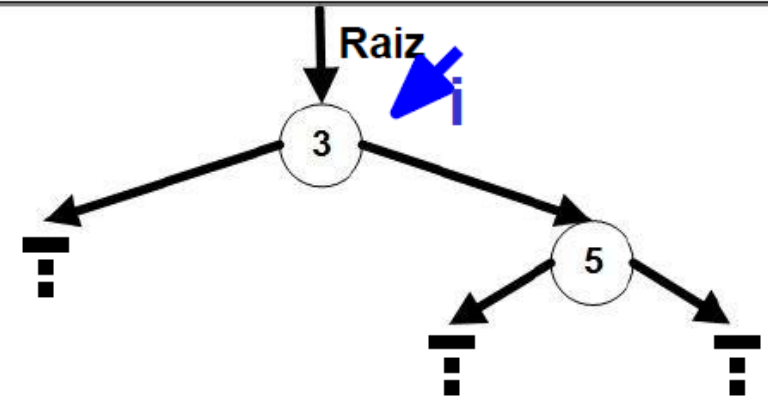
Inserção com retorno de referência

//Inserir 3, **5**, 1, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 5 x 5 i n(3)



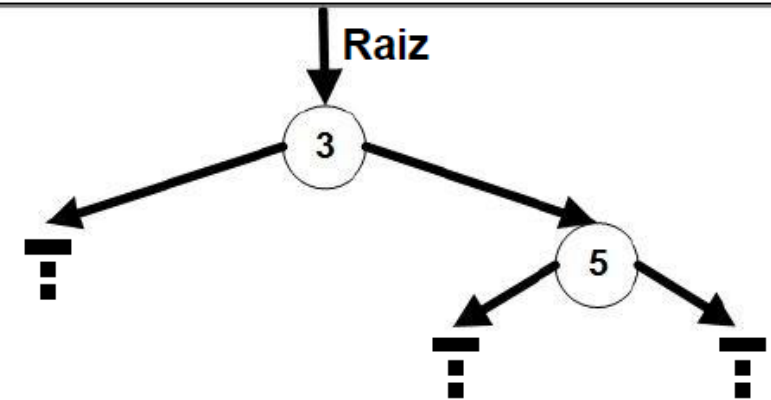
Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz $n(3)$ \times 5



Inserção com retorno de referência

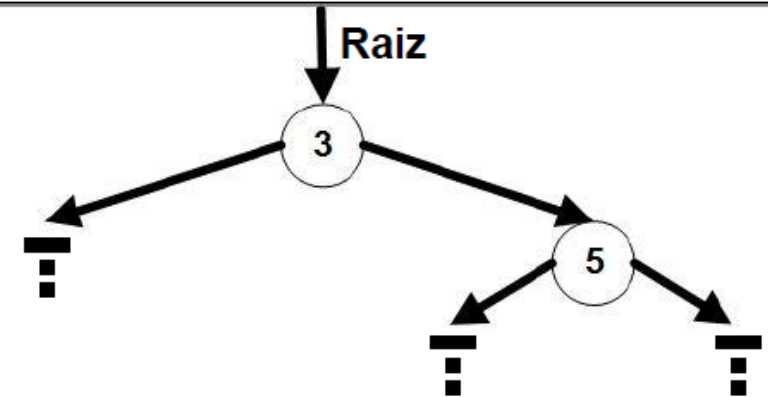
```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);  
}
```

```
private No Inserir(int x, No i){  
    if (i == null){  
        i = new No(x);  
    } else if (x < i.Elemento){  
        i.Esq = Inserir(x, i.Esq);  
    } else if (x > i.Elemento){  
        i.Dir = Inserir(x, i.Dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz

n(3)



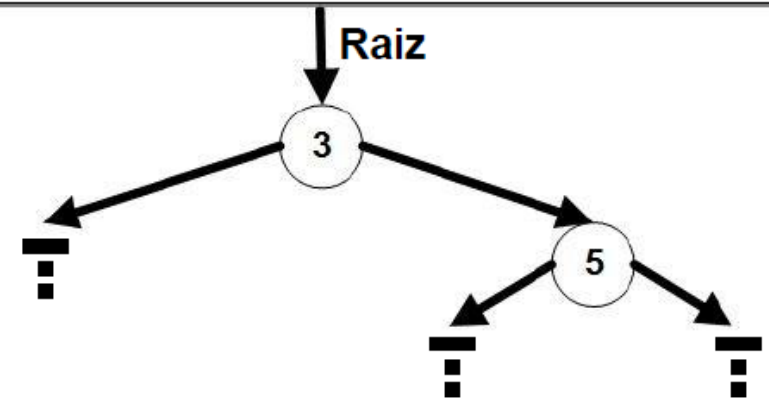
Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1



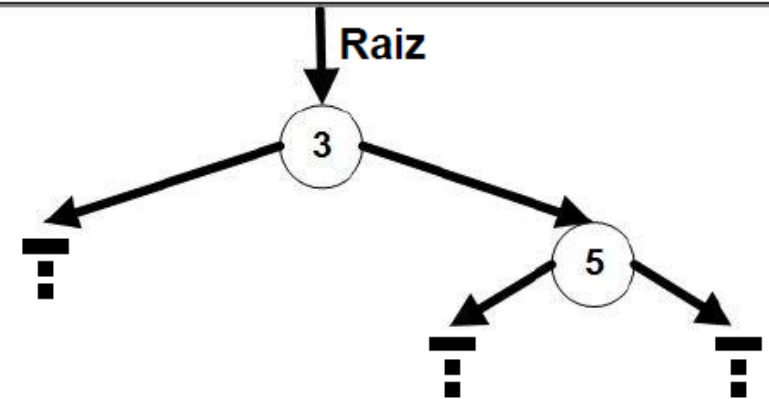
Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);  
}
```

```
private No Inserir(int x, No i){  
    if (i == null){  
        i = new No(x);  
    } else if (x < i.Elemento){  
        i.Esq = Inserir(x, i.Esq);  
    } else if (x > i.Elemento){  
        i.Dir = Inserir(x, i.Dir);  
    } else {  
        throw new Exception("Erro!");  
    }  
    return i;  
}
```

raiz $n(3)$ \times 1



Inserção com retorno de referência

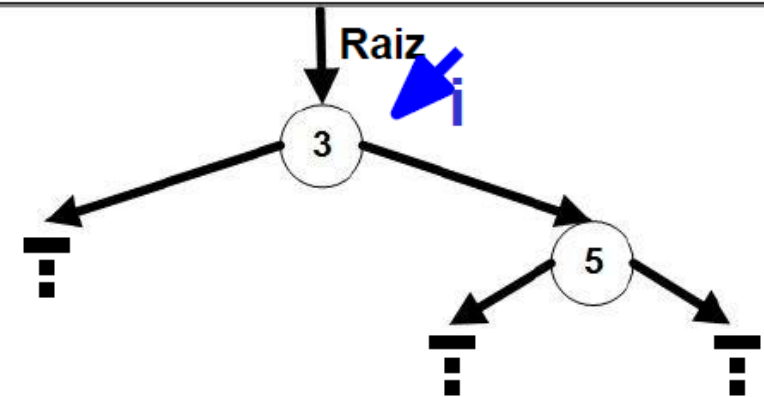
//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

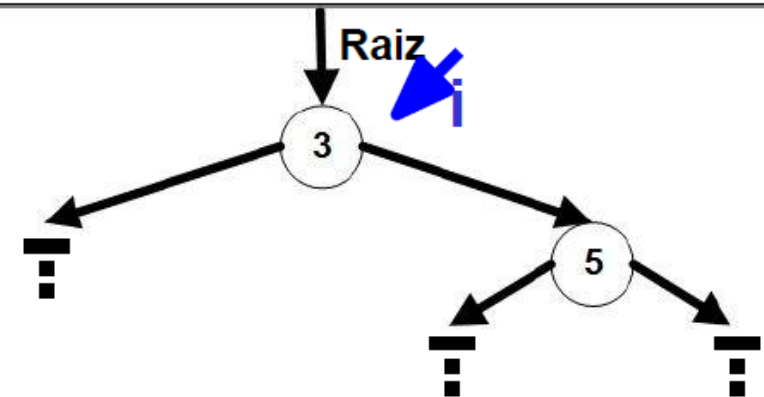
```
    }
```

```
    return i;
```

```
}
```

false: n(3) == null

raiz n(3) x 1 x 1 i n(3)



Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

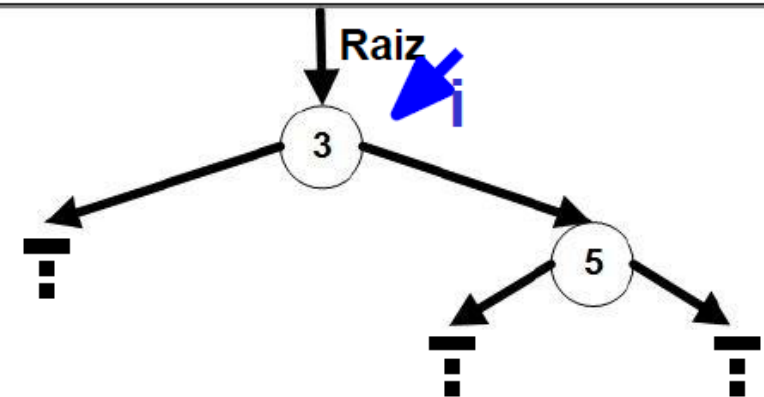
```
    }
```

```
    return i;
```

```
}
```

true: $1 < 3$

raiz n(3) x 1 x 1 i n(3)



Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

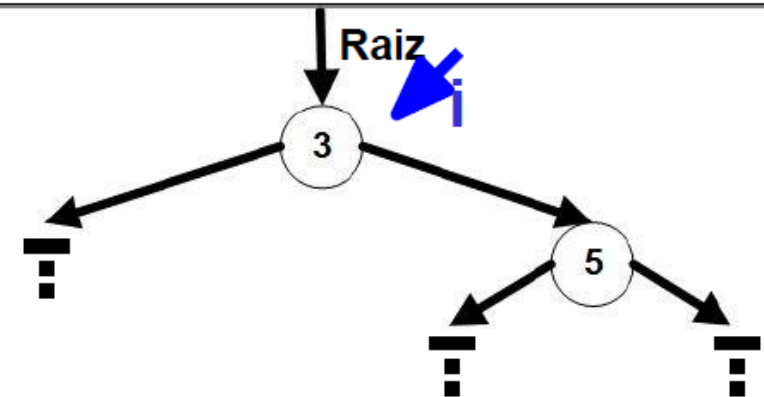
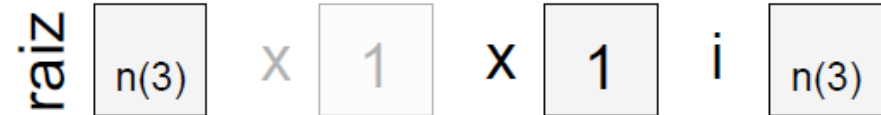
```
    } else {
```

```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```



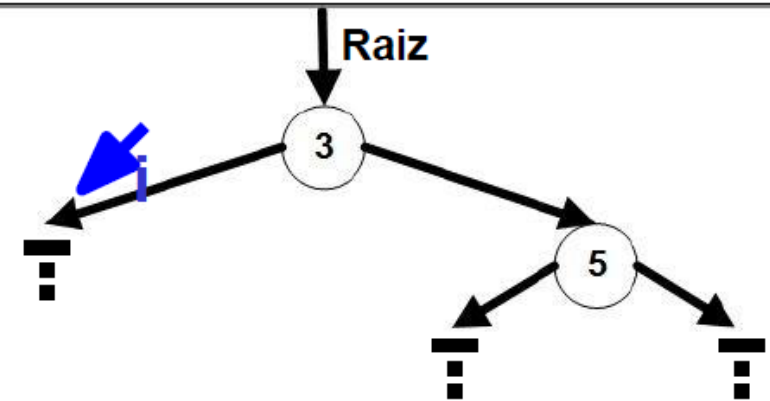
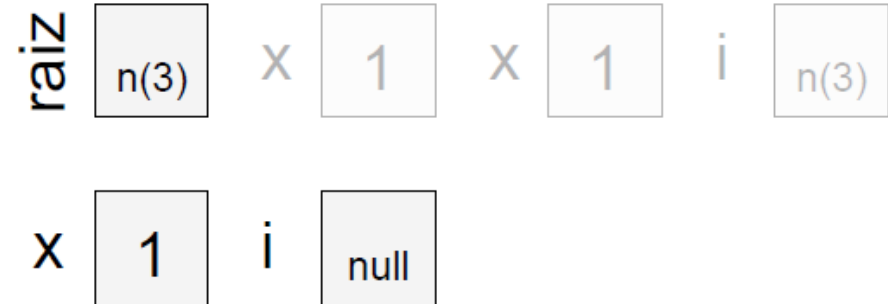
Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```



Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Insrerir(int x, No i){
```

```
if (i == null){
```

```
i = new No(x);
```

```
} else if (x < i.Elemento){
```

```
i.Esq = Inserir(x, i.Esq);
```

```
} else if (x > i.Elemento){
```

```
i.Dir = Insérer(x, i.Dir);
```

```
} else {
```

```
throw new Exception("Erro!");
```

}

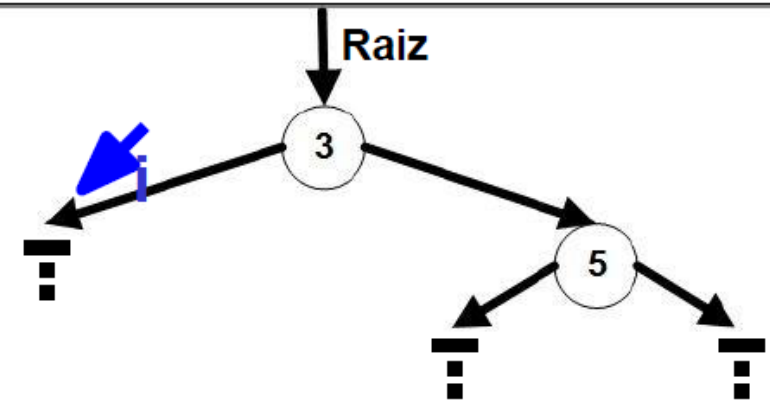
```
return i;
```

}

```
true: null == null
```

$$\text{raiz} \quad \boxed{n(3)} \quad \times \quad \boxed{1} \quad \times \quad \boxed{1} \quad i \quad \boxed{n(3)}$$

x	1	i	null
---	---	---	------



Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}
```

```
private No Inserir(int x, No i){
    if (i == null){
```

```
        i = new No(x);
```

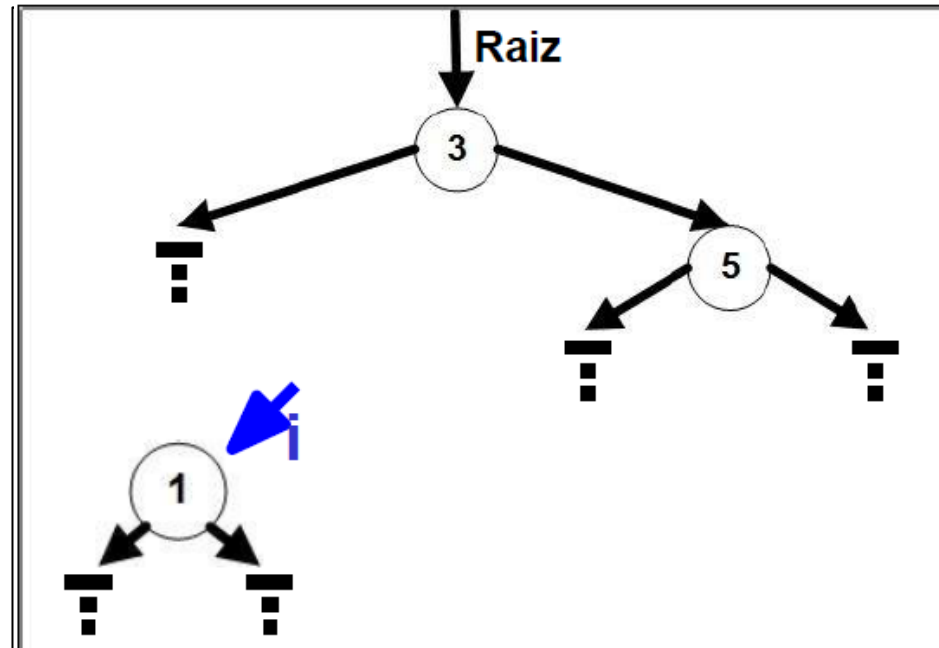
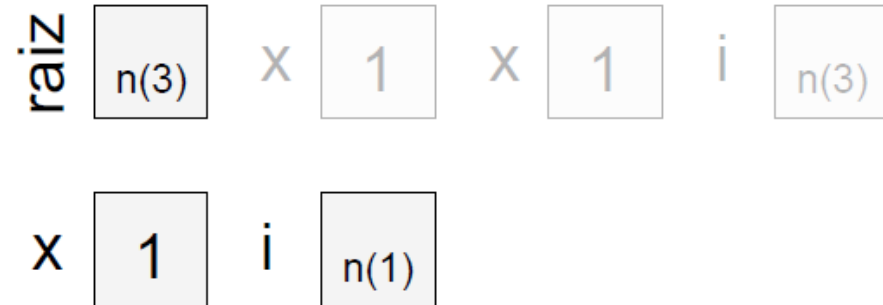
```
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
        throw new Exception("Erro!");
```

```
    }
    return i;
```

```
}
```



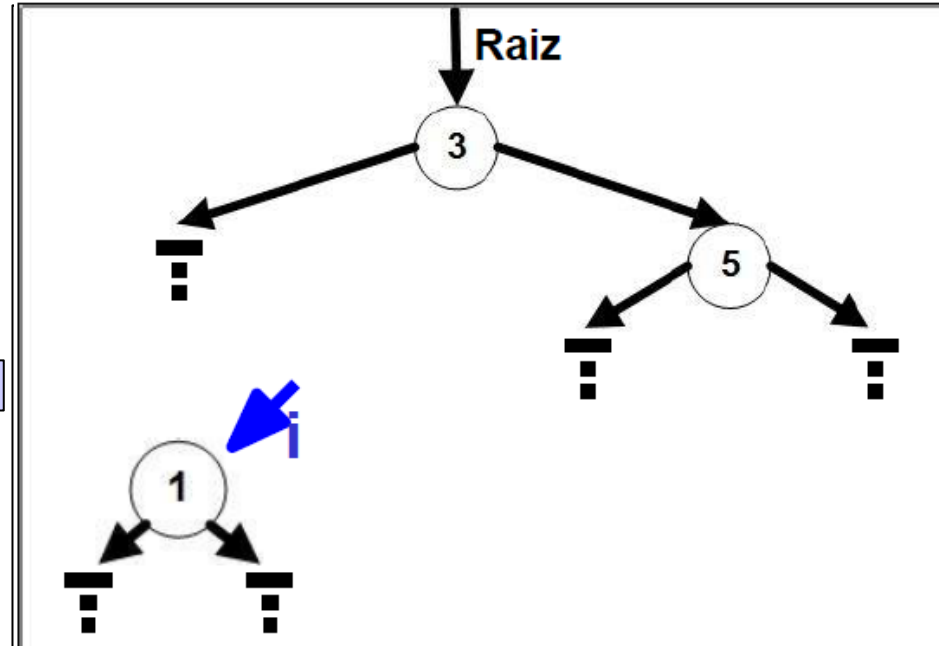
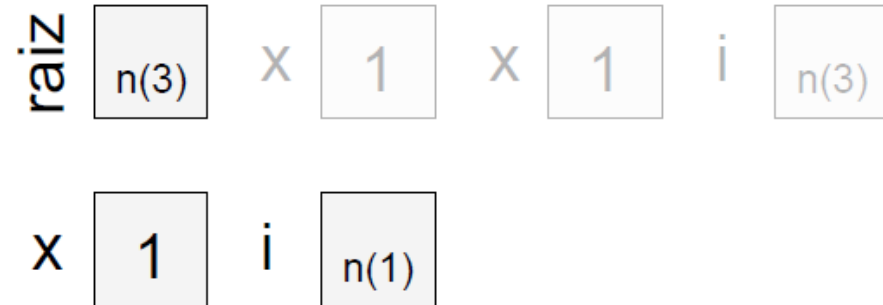
Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

retorna o endereço de n(1)

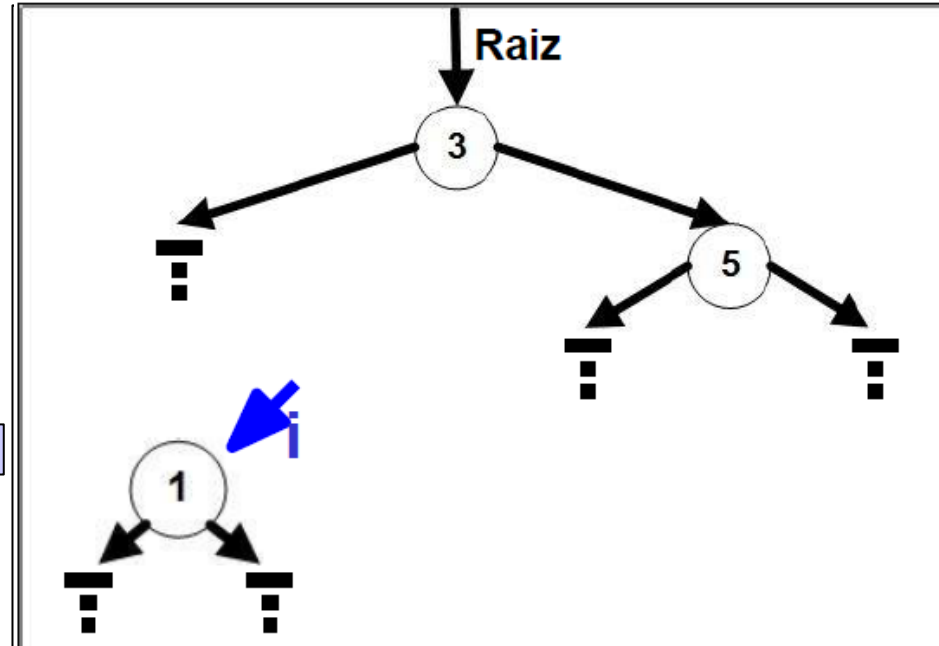
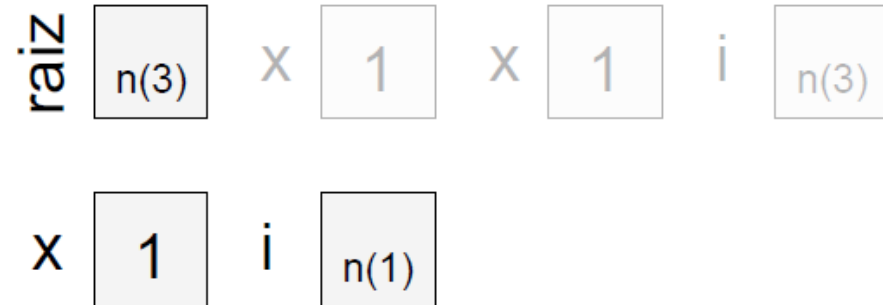


Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

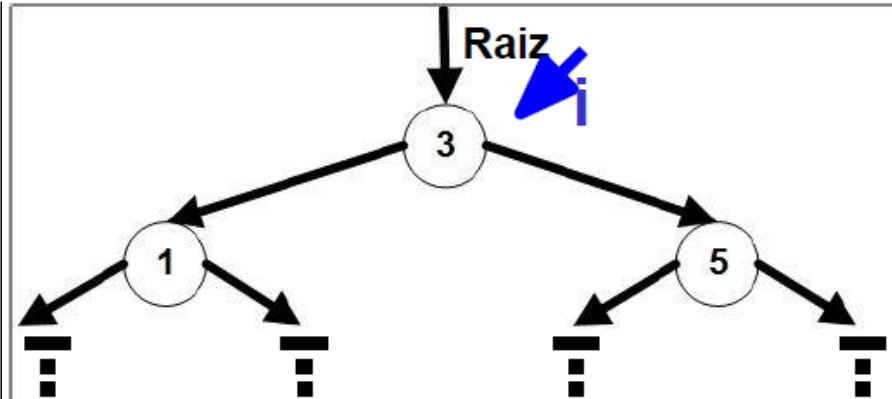
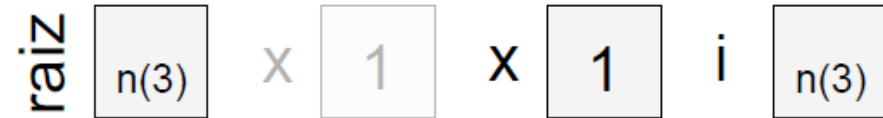


Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```



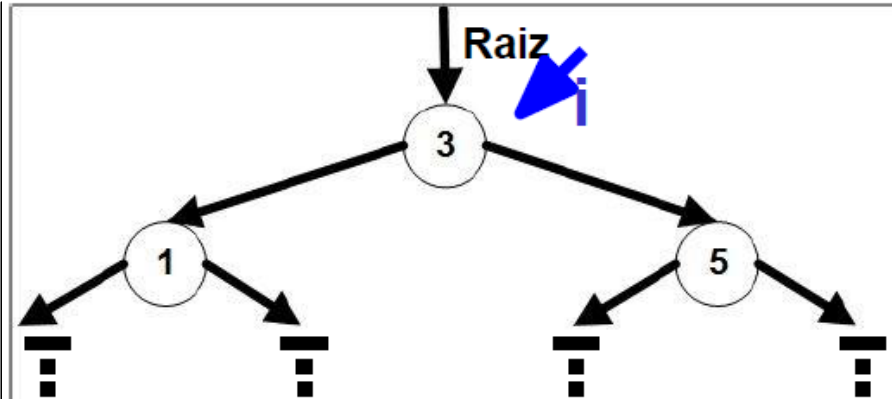
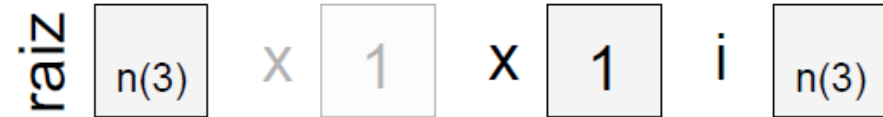
Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

retorna o endereço de n(3)



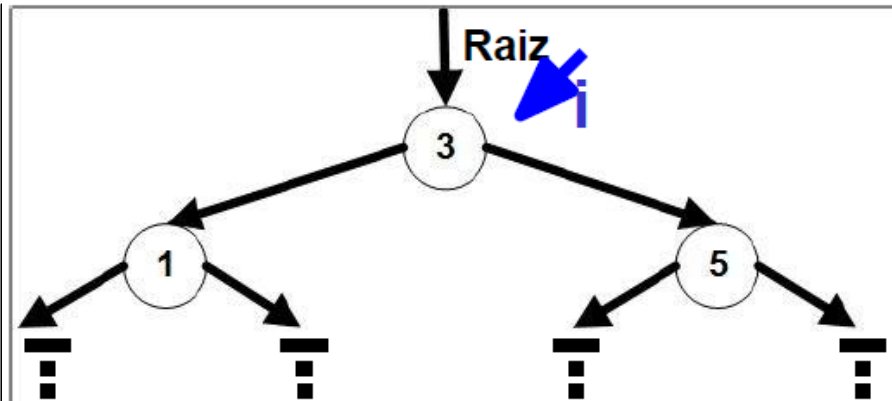
Inserção com retorno de referência

//Inserir 3, 5, **1**, 8, 2, 4, 7 e 6

```
public void Inserir(int x){
    raiz = Inserir(x, raiz);
}

private No Inserir(int x, No i){
    if (i == null){
        i = new No(x);
    } else if (x < i.Elemento){
        i.Esq = Inserir(x, i.Esq);
    } else if (x > i.Elemento){
        i.Dir = Inserir(x, i.Dir);
    } else {
        throw new Exception("Erro!");
    }
    return i;
}
```

raiz n(3) x 1 x 1 i n(3)



Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){
```

```
    raiz = Inserir(x, raiz);
```

```
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

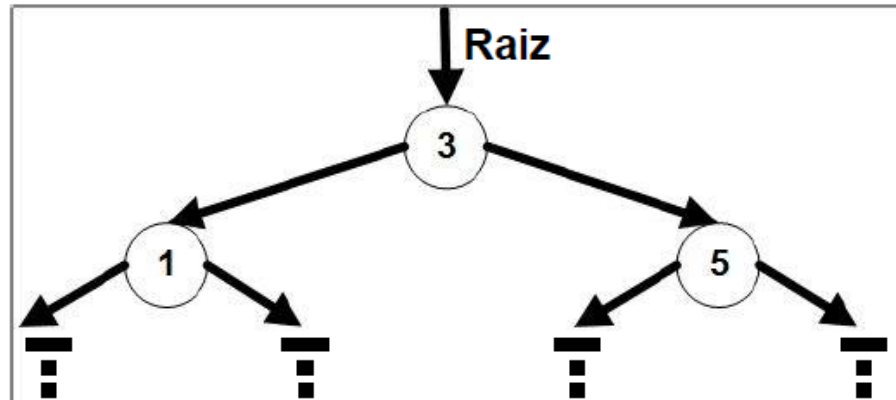
```
        throw new Exception("Erro!");
```

```
    }
```

```
    return i;
```

```
}
```

raiz n(3) X 1



Inserção com retorno de referência

```
//Inserir 3, 5, 1, 8, 2, 4, 7 e 6
```

```
public void Inserir(int x){  
    raiz = Inserir(x, raiz);
```

```
}
```

```
private No Inserir(int x, No i){
```

```
    if (i == null){
```

```
        i = new No(x);
```

```
    } else if (x < i.Elemento){
```

```
        i.Esq = Inserir(x, i.Esq);
```

```
    } else if (x > i.Elemento){
```

```
        i.Dir = Inserir(x, i.Dir);
```

```
    } else {
```

```
        throw new Exception("Erro!");
```

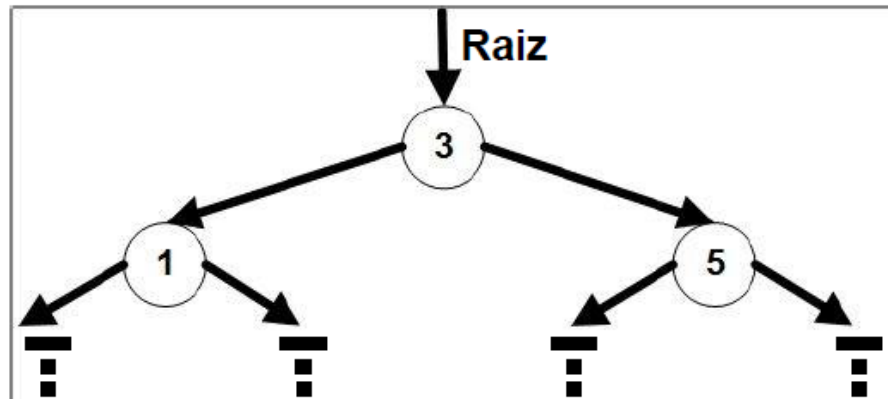
```
    }
```

```
    return i;
```

```
}
```

raiz

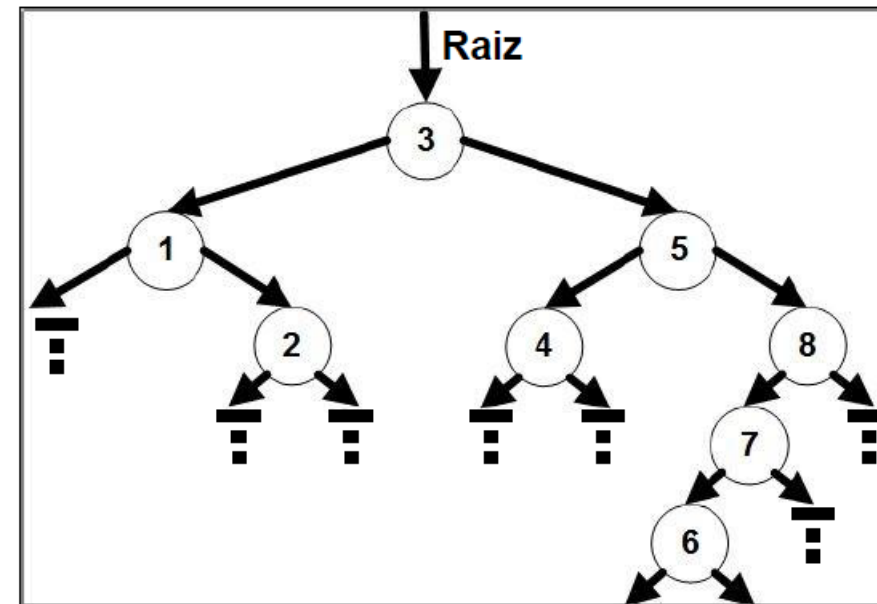
n(3)

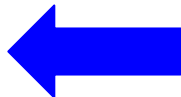


Inserção com retorno de referência

```
class ArvoreBinaria
{
    private No raiz;
    public ArvoreBinaria(){raiz = null;}
    public void Inserir(int x) { }
    public bool Pesquisar(int x) { }
    public void CaminharCentral() { }
    public void CaminharPre() { }
    public void CaminharPos() { }
    public void Remover(int x) { }
}
```

Após a inserção do 8, 2, 4, 7 e 6, temos:



- Funcionamento básico
- Exemplo
- Inserção com retorno de referência
- **Análise de complexidade** 

Análise de Complexidade da Inserção

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, inserindo na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos na ordem crescente ou decrescente
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando inserimos um elemento na folha de uma árvore balanceada. Lembrando que a altura da árvore balanceada é $\Theta(\lg(n))$

Análise de Complexidade da Inserção

- **Melhor Caso:** $\Theta(1)$ comparações e acontece, por exemplo, inserindo na raiz
- **Pior Caso:** $\Theta(n)$ comparações e acontece, por exemplo, quando inserimos os elementos na ordem crescente ou decrescente
- **Caso Médio:** $\Theta(\lg(n))$ comparações e acontece, por exemplo, quando inserimos um elemento na folha de uma árvore balanceada. Lembrando que a altura de uma árvore balanceada é $\Theta(\lg(n))$

Observação (1): Dependência do formato da árvore

Observação (2): Na inserção aleatória $\approx 1,39 \times \lg(n)$ comparações